Hedonic Game for Task Allocation in Heterogeneous Multi-Robot Systems

Hyeongseop Kim[1], Inmo Jang[2], and Hyondong Oh[1]

[1] Ulsan National Institute of Science and Technology, Ulsan, Republic of Korea,
{kimhs8090,h.oh}@unist.ac.kr,
[2] Korea Aerospace University, Goyang-si, Gyeonggi-do, Republic of Korea,
{inmo.jang}@kau.ac.kr

Abstract. This paper proposes a novel approach for task allocation in heterogeneous multi-robot systems using a hedonic game framework. The proposed method addresses the limitations of existing approaches that do not account for different levels of resources among robots with the same capability. If these differences are not considered, there can be an imbalance where some coalitions receive excessive resources while others lack sufficient resources. To address this problem, we normalize the resource with the maximum resource among all agents and incorporate it into the agent's individual preference. The proposed individual preference of each agent satisfies the single-peaked-at-one condition, which ensures the existence of Nash stable partition and convergence to it within polynomial time. We compared the proposed method with an existing approach that does not consider different levels of resources through simulations. The comparison is based on the *Balanceness* metric, the standard deviation of resource distribution among coalitions. The results demonstrated the effectiveness of considering resources in improving the overall balance of resource distribution.

Keywords: Coalition formation, heterogeneous agents, task allocation.

1 Introduction

Coalition formation has been the focus of research as multiple robots cooperate to complete complex tasks that a single robot cannot accomplish alone. This problem is typically classified as multi-robot task allocation (MRTA) and can be addressed through two main approaches: centralized, which uses global information, and distributed, which uses only local information from neighbor robots. As the number of robots increases, centralized approaches can lead to computational overload, making distributed approaches more suitable. However, distributed approaches require frequent communication between neighbor robots to reach a social agreement. This can increase the number of communications and potentially lead to communication overload.

One distributed approach is hedonic games, a game-theoretic method where each robot forms a coalition based on its individual preference to perform the

assigned tasks. The goal of hedonic games is to form stable coalitions that all robots agree upon. The key concept of Nash stability allows for a social agreement among the robots without requiring negotiations between neighbor robots, which reduces the number of communications needed.

For these reasons, various studies have been conducted applying hedonic games. Early research primarily focused on homogeneous robots, which are considered to perform identical capabilities. Jang et al. [1] proposed an autonomous decision-making framework called group agent partitioning and placing event (GRAPE) where selfish robots form coalitions through anonymous hedonic games. To reflect the characteristics of selfish robots, individual preferences were designed to satisfy the single-peak-at-one (SPAO) condition, and through this condition, the algorithm was proven to converge to Nash stability and guarantee suboptimality.

Subsequent research expanded to task allocation for heterogeneous robots, which are considered to perform different capabilities. In [2, 3], the focus was on robots with different types of sensors, such as GPS, cameras, and LiDAR, while in [4, 5], the focus was on robots that provide different services. However, these studies have a limitation in that they do not account for scenarios where robots with the same capability may have different performance levels. Even among robots that provide the same capability, as addressed in [1], differences in performance can be considered by differences in resources. For example, differences in battery capacity would fall into this category. If resource differences among agents are not considered, a coalition might receive too many resources or too few resources. This can lead to imbalances in resource distribution among coalitions.

Therefore, this paper addresses the task allocation problem by considering robots with different resources as heterogeneous robots. By combining resource differences with the utility function, agents can form coalitions with a well-balanced distribution of resources. This makes them more suitable for accomplishing all tasks. Additionally, since the preferences that reflect resource differences also satisfy the SPAO condition, the proposed method ensures convergence to Nash stability within polynomial time.

The rest of this paper is organized as follows. Section 2 explains the problem formulation, focusing on how task allocation is managed when agents have different resources, treating them as heterogeneous agents. Section 3 introduces the proposed algorithm, outlining the hedonic game-based approach and how coalitions are formed. Section 4 presents numerical simulations of the proposed algorithm in different scenarios and the comparison with the existing method that does not consider the difference in level of resources. Finally, conclusions of the paper and suggestions for future research are given in Section 5.

2 Problem Formulation

Let $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ be the set of n agents, where each agent a_i has a resource r_i, with $r_i > 0$. The set of tasks is denoted by $\mathcal{T} = \mathcal{T}^* \cup \{t_\phi\}$, where $\mathcal{T}^* =$

$\{t_1, t_2, \ldots, t_m\}$ and t_ϕ represents a void task (i.e., not performing any specific task). Since this is a multi-robot single-task (MR-ST) problem, we assume $n > m$. Let S_j be the coalition formed by agents assigned to task t_j for each $t_j \in \mathcal{T}$, including the void task t_ϕ. The set of all coalitions $\{S_1, S_2, \ldots, S_m, S_\phi\}$ forms a partition Π of the agent set $\mathcal{A}$ as:

$$\Pi = \{S_1, S_2, \ldots, S_m, S_\phi\}.$$

This partition satisfies the following conditions:

$$\bigcup_{\forall t_j \in \mathcal{T}} S_j = \mathcal{A}, \quad S_j \cap S_k = \emptyset \quad \text{for } j \neq k.$$

Each agent a_i has an individual utility $u_i(t_j, |S_j|)$ associated with the task $t_j \in \mathcal{T}$, which depends on the agent's resource r_i, the size of the coalition $|S_j|$, and the cost of performing the task. The objective of this task allocation problem is to find the partition that maximizes the global utility, which is defined as the sum of the individual utilities across all agents. The problem described above is expressed as:

$$\max_{\{x_{ij}\}} \sum_{\forall a_i \in \mathcal{A}} \sum_{\forall t_j \in \mathcal{T}} u_i(t_j, |S_j|) x_{ij},$$

subject to

$$\sum_{\forall t_j \in \mathcal{T}} x_{ij} \leq 1, \quad \forall i \in \{1, 2, \ldots, n\},$$

$$x_{ij} \in \{0, 1\}, \quad \forall a_i \in \mathcal{A}, \quad \forall t_j \in \mathcal{T},$$

where $x_{ij} = 1$ if the agent a_i is assigned to task t_j, and $x_{ij} = 0$ otherwise.

2.1 Utility Function Incorporating Normalized Resources

In GRAPE [1], if all agents are homogeneous, the task's reward can be distributed equally among the coalition members (i.e., equal distribution). Therefore, the individual utility $u_i(t_j, |S_j|)_{equal}$ for each agent a_i is defined as:

$$u_i(t_j, |S_j|)_{equal} = \frac{R_j}{|S_j|} - c_i(t_j), \tag{1}$$

where R_j is the total reward for task t_j, and $c_i(t_j)$ is the cost for agent a_i to perform task t_j. This approach ensures that each agent a_i receives an equal portion of the total reward R_j, regardless of their resources.

However, when considering each agent's resource, it is necessary to adjust how the total reward R_j is distributed. We assume that agents with a greater resource contribute more to the task, so it is reasonable that they should receive a larger portion of the total reward R_j. To reflect this, the resource values are normalized, resulting in a normalized resource w_i for each agent a_i as:

$$w_i = \frac{r_i}{r_{max}},$$

where r_{max} is the maximum resource value among all agents. Then, the modified individual utility, which incorporates the resource, is defined by combining the normalized resource w_i with the utility function (1). Thus, the modified individual utility $u_i(t_j, |S_j|)$ for each agent a_i is given by:

$$u_i(t_j, |S_j|) = w_i \frac{R_j}{|S_j|} - c_i(t_j). \tag{2}$$

This modification ensures that the total reward R_j is distributed more fairly by considering each agent's resource. An agent with a smaller resource has a lower w_i, which means it contributes less to the task and therefore receives a smaller portion of the reward. Consequently, this agent needs to be more mindful of the cost associated with performing the task.

Moreover, this resource-based approach aligns with GRAPE when dealing with homogeneous agents. If all agents have identical resources, the normalized resource w_i for each agent will be 1. In this case, the modified utility function (2) simplifies back to the original utility function (1). This consistency ensures that our modified utility function is a valid generalization of GRAPE, making it applicable to both homogeneous and heterogeneous agent systems.

3 Algorithm Description

3.1 Hedonic Game Algorithm

In this section, we present the task allocation algorithm based on an anonymous hedonic game [1]. Each agent a_i chooses the coalition S_j based on its individual preference, which is determined by the individual utility (2). The individual preference relationship is defined as:

Definition 1 (Preference Relation). *An agent a_i prefers coalition S_j over coalition S_k if and only if $u_i(t_j, |S_j|) > u_i(t_k, |S_k|)$. Formally, this can be expressed as:*

$$S_j \succ_i S_k \iff u_i(t_j, |S_j|) > u_i(t_k, |S_k|). \tag{3}$$

Following the preference relation (3), each agent proceeds with the decision-making process to select its most preferred coalition. This process is conducted by using the algorithm proposed in [1]. In the decision-making algorithm, each agent a_i initializes its local variables such as Π^i, `satisfied`, r^i, and s^i (Lines 1-2). Here, Π^i is the locally known partition of the agent; `satisfied` is a binary variable that indicates whether the agent is satisfied with Π^i or not; r^i is an integer variable representing how many times Π^i has been updated; and s^i is a uniform random variable generated when Π^i is updated (i.e., random timestamp). Given Π^i, each agent a_i that is currently unsatisfied (i.e., `satisfied` = *false*) finds the most preferred coalition, assuming that the other agents remain in their coalitions (Lines 4-5). If the agent finds a coalition that is more preferred than the current one, the agent joins the newly found coalition and updates Π^i. Additionally, the agent increments r^i and generates a new random timestamp s^i between 0 and

1 (Lines 6-10). In any case, if the agent confirms that the currently selected coalition is the most preferred, the agent becomes satisfied with the partition Π^i (Line 11). Once the decision-making process is complete, agent a_i broadcasts a message $M^i = \{r^i, s^i, \Pi^i\}$ to its neighbor agents and receives messages M^k from them (Line 13). To ensure consistency in the partition across all agents, the algorithm uses a distributed mutex subroutine (Algorithm 2) that compares local partitions and resolves any conflicts (Lines 14-15).

In the distributed mutex subroutine, each agent a_i compares the received message set $\mathcal{M}^i_{\text{rec}}$ to determine if its partition Π^i is still valid, meaning that it reflects the most up-to-date information among neighbor agents. If $r^k > r^i$ or if $r^k = r^i$ and $s^k > s^i$, it indicates that agent a_k has more recent information than agent a_i. Consequently, Π^k is considered more valid than Π^i. In this case, agent a_i updates r^i, s^i, and Π^i to r^k, s^k, and Π^k, respectively, and sets `satisfied` to *false* (Line 3-10). At this time, `satisfied` being set to *false* means that the agent needs to go through the decision-making process again and reconsider its coalition choice.

3.2 Existence of and Convergence to a Nash Stable Partition

Finding a stable solution in task allocation is essential because it ensures that agents remain in a situation where they do not want to change their assigned tasks. Therefore, it is important to prove whether a Nash stable partition exists

Algorithm 1 Decision-making algorithm for each agent a_i [1]

// Initialization
1: `satisfied` $\leftarrow$ *false*; $r^i \leftarrow 0$; $s^i \leftarrow 0$
2: $\Pi^i \leftarrow \{S_\emptyset = \mathcal{A}, S_j = \emptyset\ \forall t_j \in \mathcal{T}\}$
// Decision-making process begins
3: **while** *true* **do**
// Make a new decision if necessary
4: **if** `satisfied` = *false* **then**
5: $S_{j^*} \leftarrow \arg\max_{\forall S_j \in \Pi^i} u_i(t_j, |S_j \cup \{a_i\}|)$
6: **if** $S_{j^*} \succ_i S_{\Pi^i(i)}$ **then**
7: Join S_{j^*} and update Π^i
8: $r^i \leftarrow r^i + 1$
9: $s^i \in$ unif[0, 1]
10: **end if**
11: `satisfied` = *true*
12: **end if**
// Broadcast the local information to neighbor agents
13: Broadcast $M^i = \{r^i, s^i, \Pi^i\}$ and receive M^k from its neighbors $\forall a_k \in \mathcal{N}_i$
// Select the valid partition from all the received messages
14: Construct $\mathcal{M}^i_{\text{rev}} = \{M^i, \forall M^k\}$
15: $\{r^i, s^i, \Pi^i\}$, `satisfied` $\leftarrow$ D-MUTEX($\mathcal{M}^i_{\text{rev}}$)
16: **end while**

Algorithm 2 Distributed mutex subroutine [1]

function D-MUTEX($\mathcal{M}^i_{\text{rev}}$)
  `satisfied` $\leftarrow$ *true*
  for each message $M_k \in \mathcal{M}^i_{\text{rev}}$ **do**
    if $(r^k > r^i)$ or $(r^k = r^i \ \&\ s^k > s^i)$ **then**
      $r^i \leftarrow r^k$
      $s^i \leftarrow s^k$
      $\Pi^i \leftarrow \Pi^k$
      `satisfied` $\leftarrow$ *false*
    end if
  end for
  return $\{r^i, s^i, \Pi^i\}$, `satisfied`
end function

and converges in the context of hedonic games. First, it is necessary to introduce the definition of a Nash stable partition.

Definition 2 (Nash Stable). *A partition* Π *is said to be Nash stable if, for every agent* $a_i \in \mathcal{A}$*, it holds that*

$$S_{\Pi(i)} \succeq_i S_j \cup \{a_i\}, \quad \forall S_j \in \Pi,$$

where $\Pi(i)$ *denotes the index of the task to which agent* a_i *currently belongs.*

The existence of and convergence to a Nash stable partition are closely related to individual preferences. The individual utility (2) decreases as the size of the coalition increases, leading agents to prefer coalitions with fewer members. This behavior aligns with the SPAO condition proposed in GRAPE [1]. Because individual preferences satisfy this condition, a Nash stable partition always exists and convergence to it within polynomial time is guaranteed. This means that even though we use a different utility (2), a similar argument to [1] can be used for stability and convergence proof.

4 Simulations

This section aims to evaluate the performance of the proposed utility function (2) in a heterogeneous multi-robot system by comparing it with the utility function (1) for equal distribution. The objective is to demonstrate that the proposed utility function results in a more balanced coalition formation, particularly when agents have different resources.

4.1 Simulation Setting

The simulation environment is set within a 500×500 space for tasks and a 300×300 space for agents. In this space, 5 tasks and 160 agents are randomly generated. Each agent is assigned a resource value, uniformly distributed between

0.1 and 1. The total reward R_j of each task, which is directly related to the task's demand (i.e., the amount of resources required to complete the task), is uniformly distributed between $1000 \times \frac{n}{m}$ and $1500 \times \frac{n}{m}$. The communication network among agents is established based on their relative distances. A communication matrix is generated where two agents can communicate if the distance between them is less than or equal to 100. The resulting communication network is strongly connected, ensuring that information can be shared across the entire network of agents. This simulation setting is executed 100 times through Monte Carlo simulations. An example of the simulation setup is shown in Fig. 1.

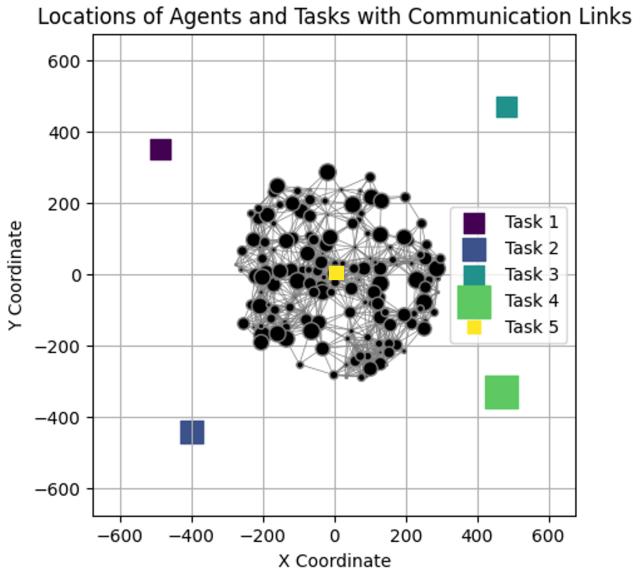
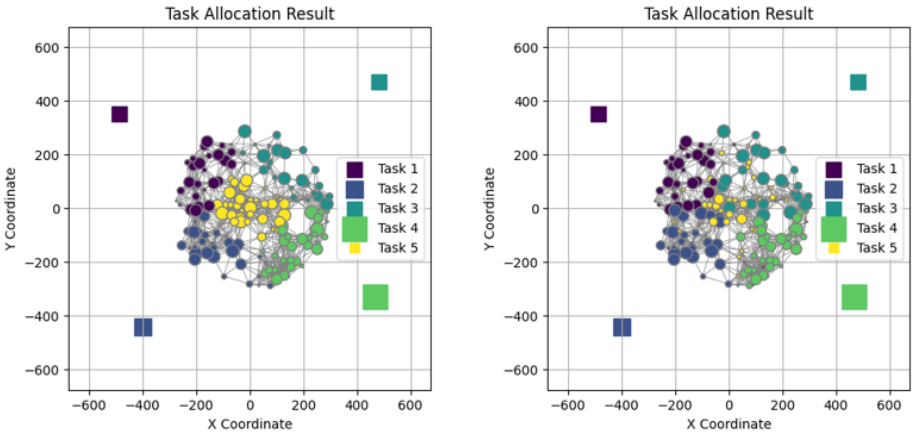


Fig. 1. An example of the simulation: agents are represented by circles sized according to their resources, tasks are represented by squares sized according to their rewards, and gray lines indicate communication links.

4.2 Simulation Results and Discussion

The task allocation results for both the equal distribution method and the proposed method are shown in Fig. 2. In the equal distribution method, agents tend to form groups mainly based on how close they are to the tasks. This means that many agents end up being assigned to tasks that are nearby, even if those tasks offer smaller rewards. For example, Task 5, which has a smaller reward, still attracts many agents because it is close to them in Fig. 2(a). On the other hand, the method considers not just the distance but also the resources that each agent has when forming coalitions. This allows agents with more resources to be

assigned to tasks with higher rewards, even if these tasks are further away, as shown in Fig. 2(b).



(a) The equal distribution method **(b)** The proposed method

Fig. 2. Comparison of task allocation results using The equal distribution method and the proposed method.

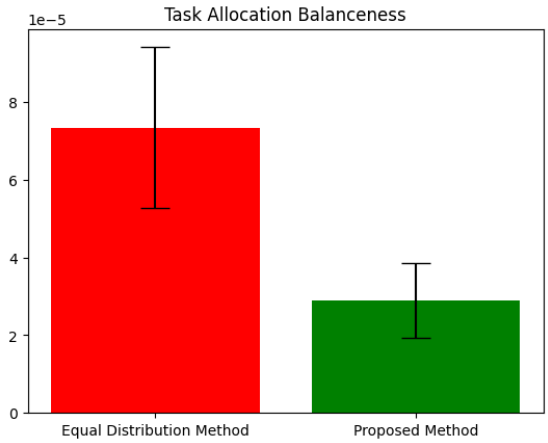


Fig. 3. Comparison of *Balanceness* between the equal distribution method and the proposed method.

To evaluate how well distributed the resources are balanced across coalitions, we define the metric called *Balanceness* as:

$$Balanceness = \sigma\left(\frac{\sum_{a_i \in S_j} w_i}{R_j}\right), \quad \forall t_j \in \mathcal{T},$$

where $\sigma(\cdot)$ is the standard deviation of the ratios of the total allocated resources to the total reward R_j for each task. *Balanceness* indicates how well the resources are balanced according to the task demands, which are related to the task rewards. Therefore, a lower *Balanceness* implies that the resources have been spread out more evenly across all tasks, ensuring that the coalitions formed are better aligned with the task requirements. In Fig. 3, the average *Balanceness* calculated over 100 simulations is presented for both the proposed method and the equal distribution method. The results show that the proposed method achieves a significantly lower *Balanceness* compared to the equal distribution method. This indicates that our approach leads to a more balanced resource allocation. Therefore, considering resources in task allocation is more effective in forming coalitions where resources are distributed more evenly, ensuring that the allocation aligns better with the task demands.

5 Conclusions and Future Work

This paper presented a distributed task allocation algorithm based on a hedonic game for agents with different levels of resources. By normalizing each agent's resource and incorporating this normalized value into individual preferences, our approach ensures a fairer distribution of resources across the multi-agent system, leading to more effective task allocation. The proposed algorithm also guarantees convergence to a Nash stable partition, satisfying the single-peaked-at-one (SPAO) preference condition. Also, simulation results showed that the proposed method leads to a more balanced distribution of resources across tasks compared with the equal distribution method.

For future work, we plan to evaluate the proposed algorithm in more complex and dynamic environments, such as scenarios where agents are in motion or tasks are dynamically added. Additionally, we will explore its application to real-world heterogeneous multi-robot systems.

Acknowledgments

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2020R1A6A1A03040570) and National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2023R1A2C2003130).

References

1. Jang, I., Shin, H.S., Tsourdos, A.: Anonymous hedonic game for task allocation in a large-scale multiple agent system. IEEE Transactions on Robotics **34**(6) (2018) 1534–1548
2. Diehl, G., Adams, J.A.: Grape-s: Near real-time coalition formation for multiple service collectives. arXiv preprint arXiv:2310.12480 (2023)
3. Zhang, L., Li, M., Yang, W., Yang, S.: Task allocation in heterogeneous multi-robot systems based on preference-driven hedonic game. In: IEEE International Conference on Robotics and Automation (ICRA). (2024) 8967–8972
4. Dutta, A., Ufimtsev, V., Said, T., Jang, I., Eggen, R.: Distributed hedonic coalition formation for multi-robot task allocation. In: IEEE 17th International Conference on Automation Science and Engineering (CASE). (2021) 639–644
5. Wang, L., Qiu, T., Pu, Z., Yi, J., Zhu, J., Yuan, W.: Hedonic coalition formation for distributed task allocation in heterogeneous multi-agent system. International Journal of Control, Automation and Systems **22**(4) (2024) 1212–1224